INTERNATIONAL SEARCH REPORT

International Application No
PCT/IL 00/00769

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	EP 0 701 202 A (INT COMPUTERS LTD) 13 March 1996 (1996-03-13) abstract page 2, line 25 - line 35 page 3, line 1 -page 4, line 3 page 6, line 39 -page 7, line 5 --- -/--	1-7,10, 12,13,19 8,9,11, 14-18

AA

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 February 2002	04/03/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Archontopoulos, E

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IL 00/00769

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	COLLET C., BRUNEL É.: "DÉFINITION ET MANIPULATION DE FORMULAIRES AVEC FO2" TECHNIQUE ET SCIENCE INFORMATIQUES, AFCET, PARIS, FR, vol. 10, no. 2, 1991, pages 97-124, XP000297686 ISSN: 0752-4072 abstract page 99, left-hand column, line 16 - line 29 page 99, right-hand column, line 5 - line 34 page 100, left-hand column, line 31 - line 34 page 100, right-hand column, line 38 -page 101, left-hand column, line 9 ---	1-3,5, 12,19
A	HESKETH R.: "PERLY-UNIX WITH BUTTONS" SOFTWARE PRACTICE & EXPERIENCE, JOHN WILEY & SONS LTD., CHICHESTER, GB, vol. 21, no. 11, 1 November 1991 (1991-11-01), pages 1165-1187, XP000276627 ISSN: 0038-0644 abstract page 1169, line 12 -page 1170, line 2 page 1171, line 10 - line 19 page 1173, line 1 -page 1174, line 17 page 1179, line 21 - last line -----	1

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/IL 00/00769

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0701202 A	13-03-1996	AU 3044095 A	21-03-1996
		EP 0701202 A1	13-03-1996
		ZA 9506282 A	13-03-1996

AP

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organizati n
International Bureau

(43) International Publicati n Date
31 May 2001 (31.05.2001)

PCT

(10) International Publication Number
WO 01/38969 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/IL00/00769

(22) International Filing Date:
19 November 2000 (19.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
133082 22 November 1999 (22.11.1999) IL

(71) Applicant *(for all designated States except US)*: **ECI TELECOM LTD.** [IL/IL]; Hasivim Street 30, 49517 Petach-Tikva (IL).

(72) Inventors; and

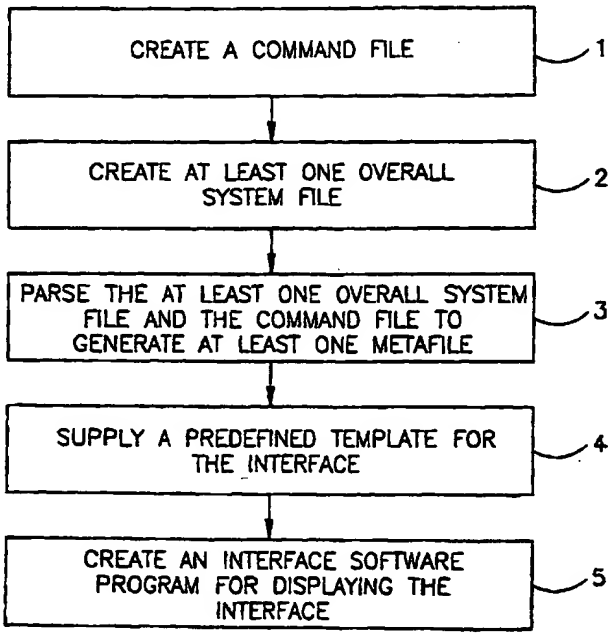
(75) Inventors/Applicants *(for US only)*: **ROZENBERG, Dominique** [IL/IL]; Ahim Kibovitch Street 7/11, 76450 Rehovot (IL). **FRUHTMAN, Felix** [IL/IL]; Beit Lehem Street 9/11, 75323 Rishon Lezion (IL). **SHAHAM, Eytan** [IL/IL]; Sorek Street 1, 48571 Rosh Ha'Ayin (IL).

(74) Agent: **INGEL, Gil**; ECI Telecom Ltd., Patent & Trademark Dept., Hasivim Street 30, 49517 Petach-Tikva (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: METHOD FOR AUTOMATICALLY CONSTRUCTING A COMMAND INTERFACE



(57) Abstract: A method for enabling a command interface for a network element to be automatically constructed from a modular command description, such that interlinking command interfaces can be created from a plurality of command descriptions. Since each command description is modular, interfaces for new commands can be easily added and integrated into an existing system of interfaces for interacting with the network element. The method enables a set of interfaces to be more easily constructed and updated for a complex system such as a T::DAX® digital cross-connect. Furthermore, these interfaces present commands grouped in a logical manner, such that each interface only shows the user commands which are available at each stage of the interaction, thereby simplifying the complexity of the management of these commands. Thus, the method enables a set of interfaces to be automatically created and updated, which both simplifies the task of programming these interfaces and of operating the interfaces once they have been created.

WO 01/38969 A2

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD FOR AUTOMATICALLY CONSTRUCTING A COMMAND INTERFACE

FIELD AND BACKGROUND OF THE INVENTION

The present invention relates to a method for constructing an interface for a network device and/or set of network elements for operation by a human user, and in particular, for such a method in which the interface is at least semi-automatically constructed by a software program, thereby significantly reducing the amount of intervention required by a human programmer.

Local area networks (LAN) are widely used for communication between computers which are relatively physically close to each other, or "local". Depending upon the type of network protocol according to which network traffic is directed, various types of hardware devices are used to physically connect each computer to a transport medium, such as a wire. The computers can then communicate by exchanging data. For example, in a client-server system, a plurality of clients could request services from a server. In order for the devices on a LAN to communicate with computers or other devices which are physically farther away, the LAN must be connected to a WAN (wide area network), the Internet or some other type of network.

All of these connections require specialized hardware devices, known as network elements. These network elements must be recognized and managed by network management systems. The interaction of the user with such complex systems requires the user to select and execute commands in a logical manner, such that values for parameters for each command are permissible and such that the commands themselves are executed in an order which is both logical and suitable for the complex system. Therefore, the user must be presented with a GUI (graphical user interface) which only enables permitted commands to be selected, and which prevents the user from entering forbidden and/or illogical values for the parameters for each command.

According to the background art, each such interface must be designed and implemented manually, by a human programmer. However, this solution is very time consuming and tedious, as it requires substantially all of the work of implementation to be performed by the human programmer. Furthermore, adding new commands to such a manually constructed interface is a difficult and time consuming process, as these new commands must also be added manually.

Figure 1 shows an exemplary GUI for operating a particularly complex network element system, known as a T::DAX® digital cross connect. The T::DAX® digital cross-connect device is like a switch. According to one functionality thereof, an incoming signal is received from a so-called "from port", and an outgoing signal is transmitted through a

so-called "to port". The cross-connect must be defined in order to pass the signal from the "from port" to the "to port". The number of elements in a typical T::DAX® digital cross-connect is very large, even up to thousands of network elements, such that manual operation of the associated commands is extremely difficult and in some cases almost impossible. Therefore, a GUI (graphical user interface) is required in order to assist the user to manage the operation of the T::DAX® digital cross-connect.

According to the background art, a GUI **10** enables the user to select a type of network element **12**, a location of the element **14** and a type of operation **16**. Once these choices have been made, then the user may select one of a plurality of commands from a command menu **18**. After selecting the command, the user then must enter values for various parameters in a parameter entry sub-window **20**, shown here with a mixture of check boxes, pull-down menus and manual entry boxes, it being understood that the mixture of elements for entering parameter values would clearly depend upon the parameters themselves.

The significant drawback of background art methods for constructing GUI **10** is that parameter entry sub-window **20** has to be built separately and manually for each command, clearly a repetitive and tedious task. Furthermore, each type of operation **16** must be programmed separately. Since these tasks are repetitive, a significant amount of time is wasted for constructing GUI **10**.

A more useful solution would be generic to commands for any

complex system, and would be built in a modular fashion, such that new commands could be easily accommodated. At the minimum, such a solution would enable at least a portion of the interface to be constructed automatically, thereby reducing the amount of work required for the human programmer. Furthermore, the solution would also organize the commands, thereby producing a GUI display which is simpler to use, as only relevant and suitable commands would be displayed to the user in each screen. Unfortunately, such a solution is not currently available.

Therefore, there is an unmet need for, and it would be highly useful to have, a method for automatically constructing a GUI for interacting with a complex system, which is able to parse information concerning each individual command and to determine the logical relationships between commands, and which is therefore more efficient.

SUMMARY OF THE INVENTION

The method of the present invention enables an interface for operating a complex network system to be automatically constructed, by constructing a screen for each command. The commands are read from a command file. Additional files specify the relationships between commands, including commands which can be performed for each operation; values which are excluded and which therefore cannot be entered by the user; related help files for providing assistance to the user concerning these commands; parameter names which are network elements; and at least one command file

concerning specific information for each command. These files are fed into a parser, which generates at least one metafile for specifying the structure of the GUI (graphical user interface) to be generated for each command. The interfaces are then linked to form the overall interface for the plurality of commands. Thus, the method of the present invention enables an interface for a complex system to be generated from a plurality of modular command interfaces.

According to the present invention, there is provided a method for automatically generating an interface software program for entering a plurality of commands for operating a complex system by a user, the complex system featuring a plurality of components each supporting a sub-plurality of the commands, the steps of the method comprising:

(a) providing a component interface for each of the plurality of components, the component interface including a command description of each of the corresponding sub-plurality of commands, including a description of each parameter for each command;

(b) parsing the command description from the component interface to form a description of a command interface for each command for display to the user;

(c) building the command interface for each command; and

(d) creating the interface software program for displaying the command interface for each command to the user.

It should be noted that step (a) may be considered as a step of

obtaining a ready "modular" component interface being suitable for combining with other "modular" component interfaces (for example, if all the components are network elements with TL1 interfaces). In such a module interface, relationship between the commands and the components, as well as between the commands and operations is known in advance.

However, if at least one component of the complex system does not have a suitable component interface, step (a) should comprise the following sub-steps for creating module interfaces for such components:

(i) providing a command description of each command supported by any such component;

(ii) providing a relationship between each said command and each of such components.

Hereinafter, the term "file" describes a particular set of data which is not necessarily present as a physically separate file or other data storage unit, but which is described as a file for the sake of convenience.

Hereinafter, the term "computing platform" refers to a particular computer hardware system or to a particular software operating system. Examples of such hardware systems include those with any type of suitable data processor, such as a computer. Hereinafter, the term "computer" includes, but is not limited to, personal computers (PC) having an operating system such as DOS, Windows™, OS/2™ or Linux; Macintosh™ computers; computers having JAVA™-OS as the operating system; and graphical workstations such as the computers of Sun Microsystems™ and Silicon

Graphics™, and other computers having some version of the UNIX operating system such as AIX™ or SOLARIS™ of Sun Microsystems™; a PalmPilot™, a PilotPC™, or any other handheld device; or any other known and available operating system. Hereinafter, the term "Windows™" includes but is not limited to Windows95™, Windows 3.x™ in which "x" is an integer such as "1", Windows NT™, Windows98™, Windows CE™ and any upgraded versions of these operating systems by Microsoft Corp. (USA).

For the present invention, a software application could be written in substantially any suitable programming language, which could easily be selected by one of ordinary skill in the art. The programming language chosen should be compatible with the computer according to which the software application is executed. Examples of suitable programming languages include, but are not limited to, C, C++ and Java. Furthermore, the functions of the present invention, when described as a series of steps for a method, could be implemented as a series of software instructions for being operated by a data processor, such that the present invention could be implemented as software, firmware or hardware.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is herein described, by way of example only, with reference to the accompanying drawings, wherein:

FIG. 1 is an illustrative background art GUI (graphical user interface)

for executing a plurality of commands for controlling a complex system; and

FIG. 2 is a flowchart of an exemplary method for constructing an interface according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The method of the present invention enables a command interface for a network element to be automatically constructed from a modular command description, such that interlinking command interfaces can be created from a plurality of command descriptions. Since each command description is modular, interfaces for new commands can be easily added and integrated into an existing system of interfaces for interacting with the network element. The method of the present invention enables a set of interfaces to be more easily constructed and updated for a complex system such as a T::DAX® digital cross-connect. Furthermore, these interfaces present commands grouped in a logical manner, such that each interface only shows the user commands which are available at each stage of the interaction, thereby simplifying the complexity of the management of these commands. Thus, the method of the present invention enables a set of interfaces to be automatically created and updated, which both simplifies the task of programming these interfaces and of operating the interfaces once they have been created.

According to one embodiment of the present invention, if each component interface for a particular network element, or component, is able

to interact with other such component interfaces, then the GUI system may be built from such component interfaces. Each component interface includes a command description of each of the plurality of commands. One example of such component interfaces which are suitable as modular building blocks is the TL1 type of interface.

Alternatively, if the component interfaces are either not available or else are not suitable as modular building blocks, then the GUI system is built by determining both the command description for each command and the relationship between commands, without reliance upon the component interfaces.

There is, of course, yet another and most practical option which constitutes a combination of the two above versions and comprises partially building the GUI system with respect to a part of the components from their component interfaces if readily available and suitable, and thereupon

completing the GUI system by determining the command description for each command and the relationship between commands and components with respect to the remaining components.

The principles and operation of the present invention may be better understood with reference to the drawings and the accompanying description.

Referring now to the drawings, Figure 2 is a flowchart of an exemplary method for creating a set of interfaces for executing a plurality of commands. This set of interfaces is also referred to herein as the "resultant

GUI". The method is explained with regard to a T::DAX® digital cross connect, the resultant background art GUI for which is shown and described above with regard to Figure 1, it being understood that this is for the purposes of description only and is not meant to be limiting in any way, as the method of the present invention would be suitable for operation with any complex system containing a plurality of commands.

As shown, in step 1, a command file is created. The command file contains information concerning at least one command, although preferably information concerning a plurality of commands is contained in the command file. The command file contains a description of the parameters for each command, default values for each parameter, and a range of permissible values for each parameter. If the command file includes information for only one command, then a plurality of command files must be created, one for each command or group of commands. The description of each parameter should also include the type of values expected for entry by the user, such as binary (off/on), a character string, a number or other description of the command parameter values.

In step 2, at least one overall system file is created, which defines relationships between different features of the system. Preferably, a plurality of separate system files are created, although as previously noted, each separate "file" describes a particular set of data which is not necessarily stored in a physically separate file or other physical data storage unit. Rather, a plurality of sets of data could be concatenated into a single physical

data storage unit, such as a single file for example. Each overall system file may be created manually, preferably with the assistance of a software program for file creation.

One such overall system file is a list of operations and commands which are permitted for each component of the complex system. For example, for a T::DAX® cross connect, each component would be a network element. In addition, this component file preferably also includes a description of the relationships between the commands and the operations, such that for each operation, a set of at least one permitted command is described. Preferably, the component file also includes a "security gate", or description of the security clearance which a user must have in order to execute each command for each component.

The next preferred overall system file is an excluded values file, which lists values which are neither shown as choices to the user through the resultant GUI, nor are these values accepted from the user from the resultant GUI. These excluded values are typically commands or values for debugging the resultant GUI and/or the operation of a component within the complex system. More preferably, the list of excluded values optionally decreases the number of permitted values from those given in the command file.

Another preferred overall system file is a list of help files, for providing assistance to the user for each command. Preferably, a help file is listed for each command and for each element type. More preferably, the

help file is made accessible to the user through the resultant GUI once a command is highlighted or otherwise indicated, by clicking or otherwise selecting an appropriate GUI element. Most preferably, a separate help file or at least separate help file information is provided for each combination of a network element type and a command.

Yet another preferred overall system file is a network elements file, which contains a list of parameter values which are actually network elements. The network elements file is created manually, as for the other overall system files.

In step 3, the overall system files and the command file are parsed in order to generate at least one, and preferably a plurality of, separate metafiles. The first such metafile is a GUI builder metafile. A GUI builder is a software program which creates a GUI from input information and which is known in the art. The GUI builder metafile is preferably suitable for interaction with a commercially available GUI builder software program, such as ILOG View™ (ILOG SA, France). The GUI builder metafile preferably features information for one command, in which groups of acceptable parameters and other information is given. More preferably, these groups of acceptable parameters are organized into blocks, with each block being identified as either a parameter or a network element type. The identification of each block is most preferably performed according to the information contained in the network elements file, as previously described. The GUI builder file is an input file to a specialized program for building

ILOG screens, such as a commercially available GUI builder software program, as previously described.

The second such metafile is a GUI application file for generating the underlying application for actually operating the GUI. The GUI application file contains the supported types of network elements for each command, preferably separated by category of command.

In addition, the GUI application file preferably has a list of parameters which may be manipulated by the user through the resultant GUI when the underlying application is being operated by the computer, or "running". The GUI application file is very similar to the previously described GUI builder metafile, except that the GUI application file includes the parameters and values for each command. These parameters are the fields displayed in the resultant GUI.

Preferably, each parameter is described with the name and type of parameter, optionally the block to which the parameter belongs if this is applicable, a default value for the parameter, and a description of the permissible values for the parameter. The description of the permissible values is optionally given either as minimum and maximum values for a numerically valued parameter, or alternatively as a list of names of acceptable values for a character string.

Preferably, the step of parsing the GUI application file includes the step of converting the organized text layout to header files for an underlying generic software program which actually runs the GUI system. These header

files are created by the software program incorporating the method of the present invention, and are built from the GUI application metafile. The header files are then read by the underlying software program for operating the resultant GUI. This step is preferred since it enables generic software to be reused for many different GUI systems.

Each header file preferably includes a list of the different types of network elements, or other components of the complex system such as the T::DAX® digital cross connect; a relationship between each type of element and a category of commands. a list of commands with a description of each command; a list of parameters for each command; and a security gate, which is security information for each command.

In step 4, a predefined template for creating the GUI is supplied. This predefined template preferably features the names of the fields of parameter values which are entered by the user, such as "from:" and "to:", as well as the basic layout of the GUI elements. In addition, the predefined template also includes words for describing each command, such as "RTRV-CRS". The words which are displayed are more preferably stored in a Data Dictionary, thereby enabling the user to change the words to be displayed. For example, the words for the command "RTRV-CRS" could be changed to display "Retrieve CRS". Furthermore, at run time preferably the list of network elements which are addressable is optionally and preferably displayed, thereby enabling the user to separately perform a command on a single network element in the system.

The actual, full T::DAX command name is RTRV-CRS-element type, such that if the element type is E1, then the command is given as "RTRV-CRS-E1", and so forth. The element type is displayed in the resultant GUI, with the command name. However, for clarity, optionally the "elementType" field is not displayed with the command name itself, but rather in a separate display.

In step 5, a GUI software program which is specific for operating the GUI system is created. The GUI software program receives the header file for each command, created as described above, in order to operate the GUI for each command.

Thus, the method of the present invention overcomes the limitations of the background art methods, by enabling a set of interfaces to be automatically created as a GUI system for operating a complex system such as a T::DAX® digital cross connect, as well as for all devices which use the TL1 interface. In order to be adapted to these other devices, the present invention need only receive a command file which describes each command and the associated parameter(s), or alternatively the previously described metafiles, as input. The method of the present invention is particularly suitable for operation with the TL1 interface, as each command of this interface features several parameters and values. Therefore, a GUI system is generally suitable and user-friendly for user interactions with the TL1 commands. The method of the present invention facilitates such a GUI system by enabling the system to be automatically created, which is more

efficient than manual creation of such a set of interfaces.

Indeed, on a more general level, the method of the present invention only requires a description of each command, including a description of the parameter(s) for each command, with a description of the relationships between commands and the components of the complex system, in order to build the GUI system as an interface for the user. Thus, the present invention is highly adaptable to various types of GUI systems for operating such complex systems.

While the invention has been described with respect to a limited number of embodiments, it will be appreciated that many variations, modifications and other applications of the invention may be made.

CLAIMS:

1. A method for automatically generating an interface software program for entering a plurality of commands for operating a complex system by a user, the complex system featuring a plurality of components each supporting a sub-plurality of the commands, the steps of the method comprising:

(a) providing a component interface for each of the plurality of components, said component interface including a command description of each of the corresponding sub-plurality of commands, including a description of each parameter for each command;

(b) parsing said command description from said component interface to form a description of a command interface for each command for display to the user;

(c) building said command interface for each command; and

(d) creating the interface software program for displaying said command interface for each command to the user.

2. The method of claim 1, wherein for at least a sub-plurality of the components, step (a) comprises the steps of:

(i) providing a command description of each command supported by any of said sub-plurality of the components;

(ii) providing a relationship between each said command and each of said sub-plurality of the components.

3. The method of claim 1 or 2, wherein the interface software program is generated for a complex system for interacting with a plurality of network elements, each network element corresponding to a component of the complex system.

4. The method of claim 3, wherein the complex system is a T::DAX® digital cross connect .

5. The method according to any one of claims 1 to 4, wherein step (a) further includes the steps of:

(iii) providing a default value for each parameter; and

(iv) providing a plurality of permissible values for each parameter.

6. The method of claim 5, wherein said plurality of permissible values includes a minimum value and a maximum value for numeric parameters, and a list of names for character string parameters.

7. The method of claim 5, wherein sub-steps (iii) and (iv) of step (a) further comprise the steps of determining said default value and said plurality of permissible values for each parameter for each of the plurality of components.

8. The method of claim 7, wherein sub-step (iv) of step (a) further comprises the step of providing a list of excluded values for each parameter of each command.

9. The method of claim 8, wherein the interface software program is generated for a complex system for interacting with a plurality of network elements, each network element corresponding to a component of the complex system and wherein step (a) further comprises the step of providing a list of at least one parameter corresponding to a network element.

10. The method of claim 2, wherein sub-step (ii) of step (a) further comprises the steps of:

(A) determining at least one operation for performing with each of the plurality of components; and

(B) determining a relationship between said at least one operation and each of the plurality of commands.

11. The method of claim 10, wherein step (a) further comprises the step of:

(C) determining a security clearance of the user required before a command is accessed by the user.

12. The method of claim 1, wherein step (a) further comprises the step of determining a help file for each of the plurality of commands, said help file including information for assisting the user for each command.

13. The method of claim 12, wherein the interface software program is generated for a complex system for interacting with a plurality of network elements, each network element corresponding to a component of the complex system and wherein said help file is provided for each combination of a network element and a command for operation with said network element.

14. The method of claim 1, wherein step (b) further comprises the step of determining a template for said interface.

15. The method of claim 14, wherein said interface is a GUI (graphical user interface) and said template features a plurality of GUI elements.

16. The method of claim 15, wherein said template features a name for each field corresponding to each parameter.

17. The method of claim 16, wherein said template features a name for each command, said name being altered according to a selection by the user.

18. The method of claim 15, wherein step (c) is performed by a GUI builder software program.

19. The method of claim 1, wherein step (d) further comprises the steps of:

(i) providing a generic interface operation software program;

(ii) generating a header file for each command; and

(iii) constructing the interface software program from said header file and said generic interface operation software program.

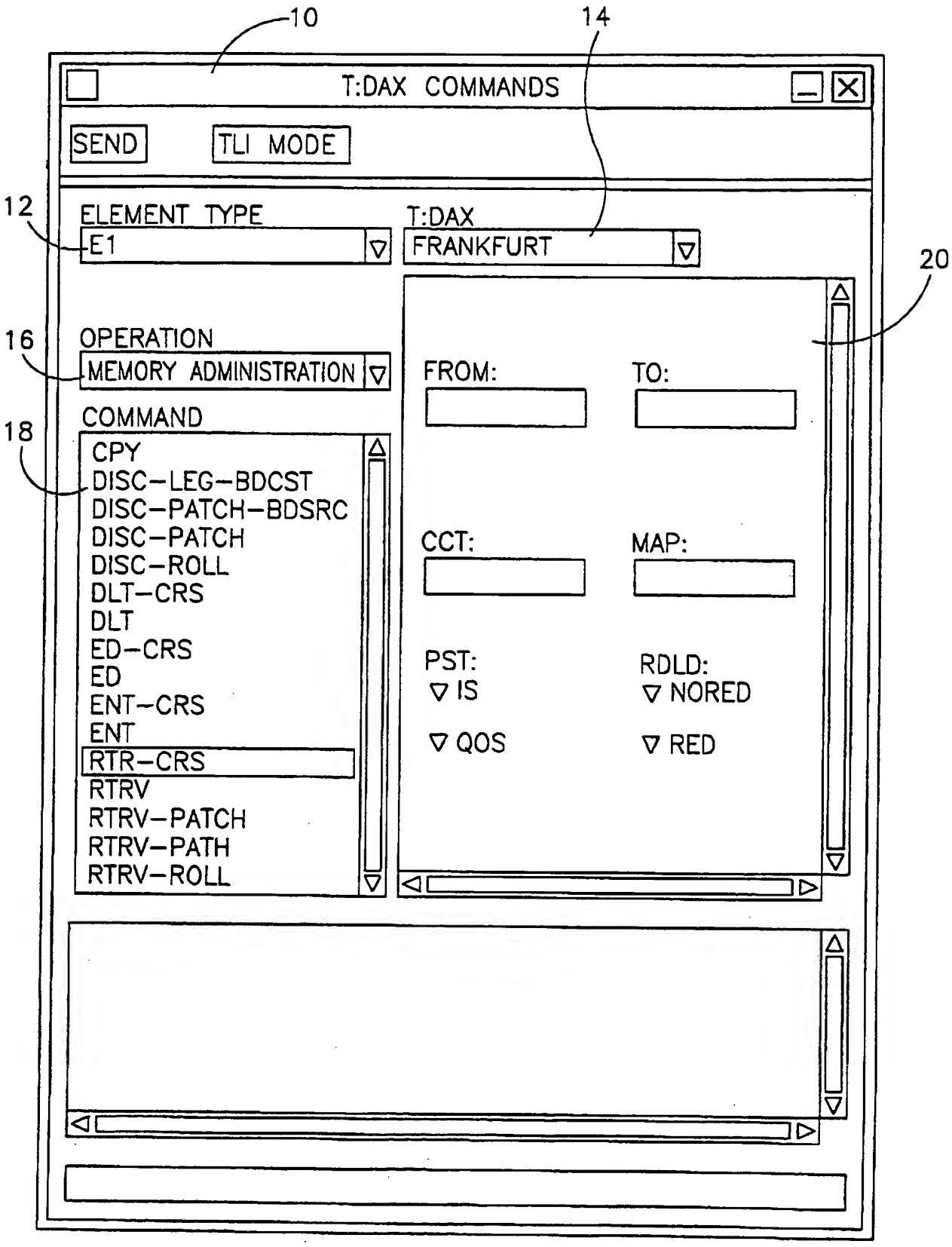

10
14
T:DAX COMMANDS
SEND
TLI MODE
12
ELEMENT TYPE
E1
T:DAX
FRANKFURT
20
16
OPERATION
MEMORY ADMINISTRATION
FROM:
TO:
18
COMMAND
CPY
DISC-LEG-BDCST
DISC-PATCH-BDSRC
DISC-PATCH
DISC-ROLL
DLT-CRS
DLT
ED-CRS
ED
ENT-CRS
ENT
RTR-CRS
RTRV
RTRV-PATCH
RTRV-PATH
RTRV-ROLL
CCT:
MAP:
PST:
IS
QOS
RDLD:
NORED
RED

FIG.1

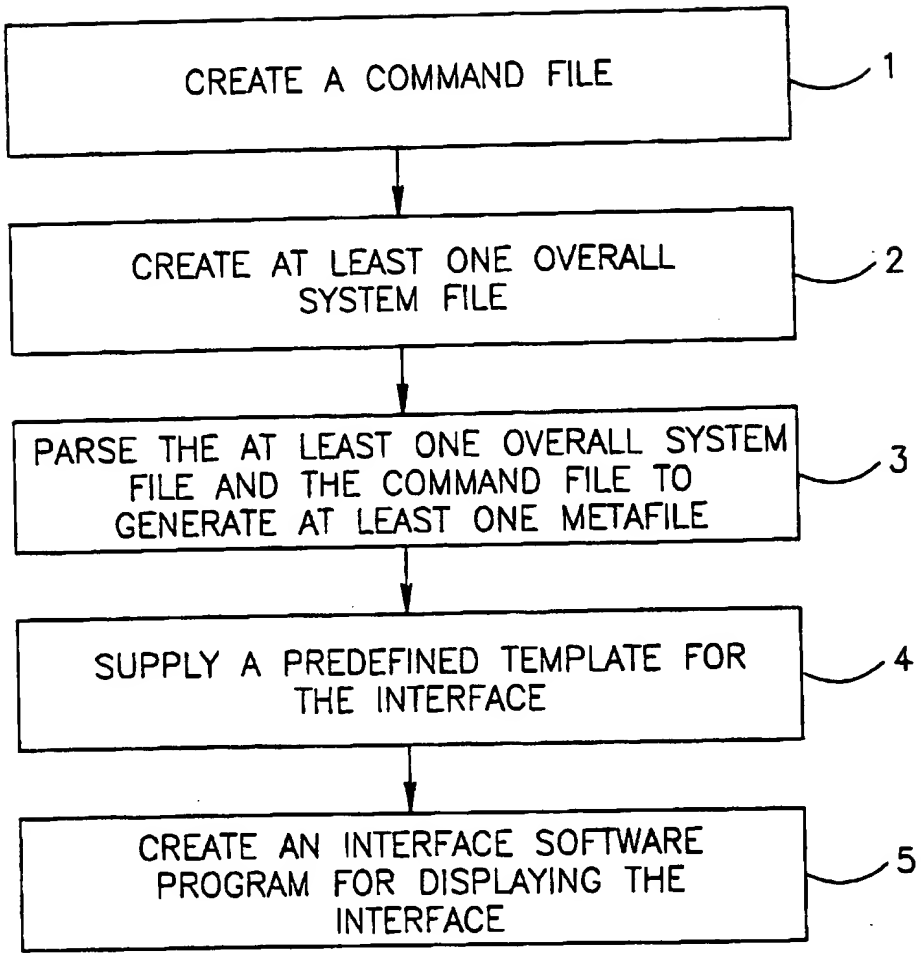

CREATE A COMMAND FILE
1
CREATE AT LEAST ONE OVERALL SYSTEM FILE
2
PARSE THE AT LEAST ONE OVERALL SYSTEM FILE AND THE COMMAND FILE TO GENERATE AT LEAST ONE METAFILE
3
SUPPLY A PREDEFINED TEMPLATE FOR THE INTERFACE
4
CREATE AN INTERFACE SOFTWARE PROGRAM FOR DISPLAYING THE INTERFACE
5

FIG.2